QBUS6850

Lecture 10

Neural Network and Deep Learning- II

BUSINESS SCHOOL

Professor Junbin Gao

THE UNIVERSITY OF SYDNEY

❑ Topics covered

- Neural Network regression and classification loss functions
- Backward propagation Neural Network
- Train a Neural Network
- Neural Network regularization

❑ References

- Alpaydin (2014), Chapter 11
- Bishop (2006), Chapter 5

Learning Objectives

- Understand the Neural Network regression and classification loss functions
- Understand why and how to add regularization terms into Neural Network
- Understand the intuition of backpropagation
- Understand the process of backpropagation
- Understand how to train a Neural Network
- Be able to build the forward propagation and backpropagation process in Python step by step and train a Neural Network with gradient descent

Neural Network Loss Function

NN Regression Loss Function

$$L(\mathbf{W}) = \frac{1}{2N}\sum_{n=1}^{N}(f(\mathbf{x}_n, \mathbf{W}) - t_n)^2$$

How NN and linear regression loss functions are different?

As in the regression we did in weeks 2&3, we can use gradient descent to try to minimize the loss function as a function of parameters. α is the learning rate.

$$\mathbf{W} \coloneqq \mathbf{W} - \alpha\frac{\partial L(\mathbf{W})}{\partial \mathbf{W}}$$

$$\frac{\partial L(\mathbf{W})}{\partial \mathbf{W}} = ?$$

This will be a much harder problem, since now we have weights $\mathbf{W}^{(1)}$, $\mathbf{W}^{(2)}$ for a 3-layer NN.

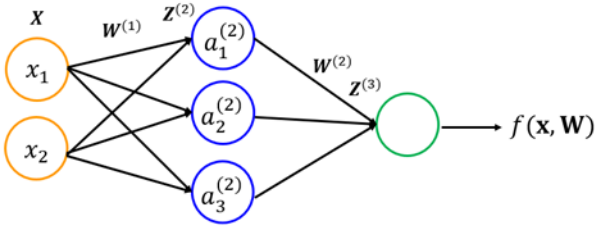
NN Classification Loss Function

Logistic regression loss function (copied from Lecture 3):

$$L(\boldsymbol{\beta}) = -\frac{1}{N}\left[\sum_{n=1}^{N}\Big(t_n \log\big(f(\mathbf{x}_n, \boldsymbol{\beta})\big) + (1 - t_n)\log\big(1 - f(\mathbf{x}_n, \boldsymbol{\beta})\big)\Big)\right]$$

where $f(\mathbf{x}_n, \boldsymbol{\beta})$ means the probability for $\mathbf{x}_n$ being class 1 ($t_n = 1$) while $1 - f(\mathbf{x}_n, \boldsymbol{\beta})$ is the probability for $\mathbf{x}_n$ being class 0 ($t_n = 0$)

TWO (binary) Classes:
NN with one output $f(\mathbf{x}, \mathbf{W})$



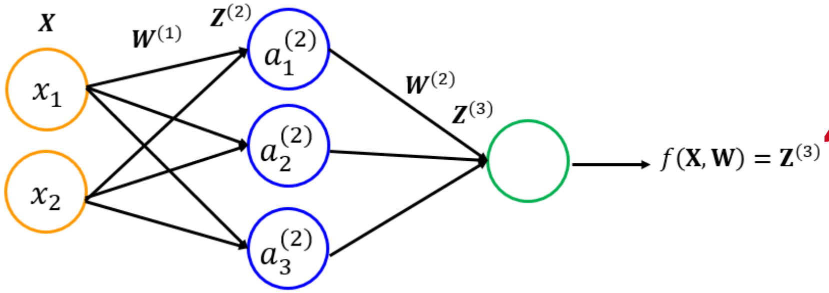
To make sure $f(\mathbf{x}, \mathbf{W})$ is probability, i.e., $0 \leq f(\mathbf{x}, \mathbf{W}) \leq 1$, we apply the sigmoid function as activation on the output neuron (Why?). Hence the loss function (cross entropy loss) is

$$L(\mathbf{W}) = -\frac{1}{N}\left[\sum_{n=1}^{N}\Big(t_n \log\big(f(\mathbf{x}_n, \mathbf{W})\big) + (1 - t_n)\log\big(1 - f(\mathbf{x}_n, \mathbf{W})\big)\Big)\right]$$

NN Classification Loss Function

TWO (binary) Classes:

Another interpretation: We don’t apply any activation at the output neuron, but use the Logistic regression loss function



No activation σ over $\mathbf{Z}^{(3)}$

Thus, the output $f(\mathbf{x}_n, \mathbf{W})$ may not be probability. Then we define the following loss (logistic loss)

$$L(\mathbf{W}) = \frac{1}{N}\sum_{n=1}^{N} \text{Loss}(f(\mathbf{x}_n,\mathbf{W}), t_n)$$

This f is the output from NN

where

$$\text{Loss}(f(\mathbf{x}_n,\mathbf{W}), t_n) = \begin{cases} -\log\left(\dfrac{1}{1+e^{-f(\mathbf{x}_n,\mathbf{W})}}\right), & t_n = 1 \\ -\log\left(1-\dfrac{1}{1+e^{-f(\mathbf{x}_n,\mathbf{W})}}\right), & t_n = 0 \end{cases}$$

NN Classification Loss Function

For class A ($t_n = +1$)

$$-\log\left(\frac{1}{1+e^{-f(\mathbf{x}_n,\mathbf{W})}}\right) = -\log\left(\frac{1}{1+e^{-(+1)f(\mathbf{x}_n,\mathbf{W})}}\right)$$

For class B ($t_n = -1$) we have

$$-\log\left(1-\frac{1}{1+e^{-f(\mathbf{x}_n,\mathbf{W})}}\right) = -\log\left(\frac{1}{1+e^{f(\mathbf{x}_n,\mathbf{W})}}\right) = -\log\left(\frac{1}{1+e^{-(-1)f(\mathbf{x}_n,\mathbf{W})}}\right)$$

If we define the target label as $t_n = +1$ (Class A) and $t_n = -1$ (Class B) , then we only need one expression for the loss function

$$\text{Loss}(f(\mathbf{x}_n,\mathbf{W}),t_n) = -\log\left(\frac{1}{1+e^{-\,t_n\,f(\mathbf{x}_n,\mathbf{W})}}\right)$$

And

$$L(\mathbf{W}) = \frac{1}{N}\sum_{n=1}^{N}\text{Loss}(f(\mathbf{x}_n,\mathbf{W}),t_n)\,, \qquad t_n = +1\; or -1$$

NN Classification-multiple output units

- NN for multiple classes: We assume K output units
- $f_k(\mathbf{x}_n, \mathbf{W})$ represents the k_{th} output unit (either applying an activation or no activation. Most time no activation)

- Now we add one more layer to transform K outputs $f_k(\mathbf{x}_n, \mathbf{W})$ to probabilities by

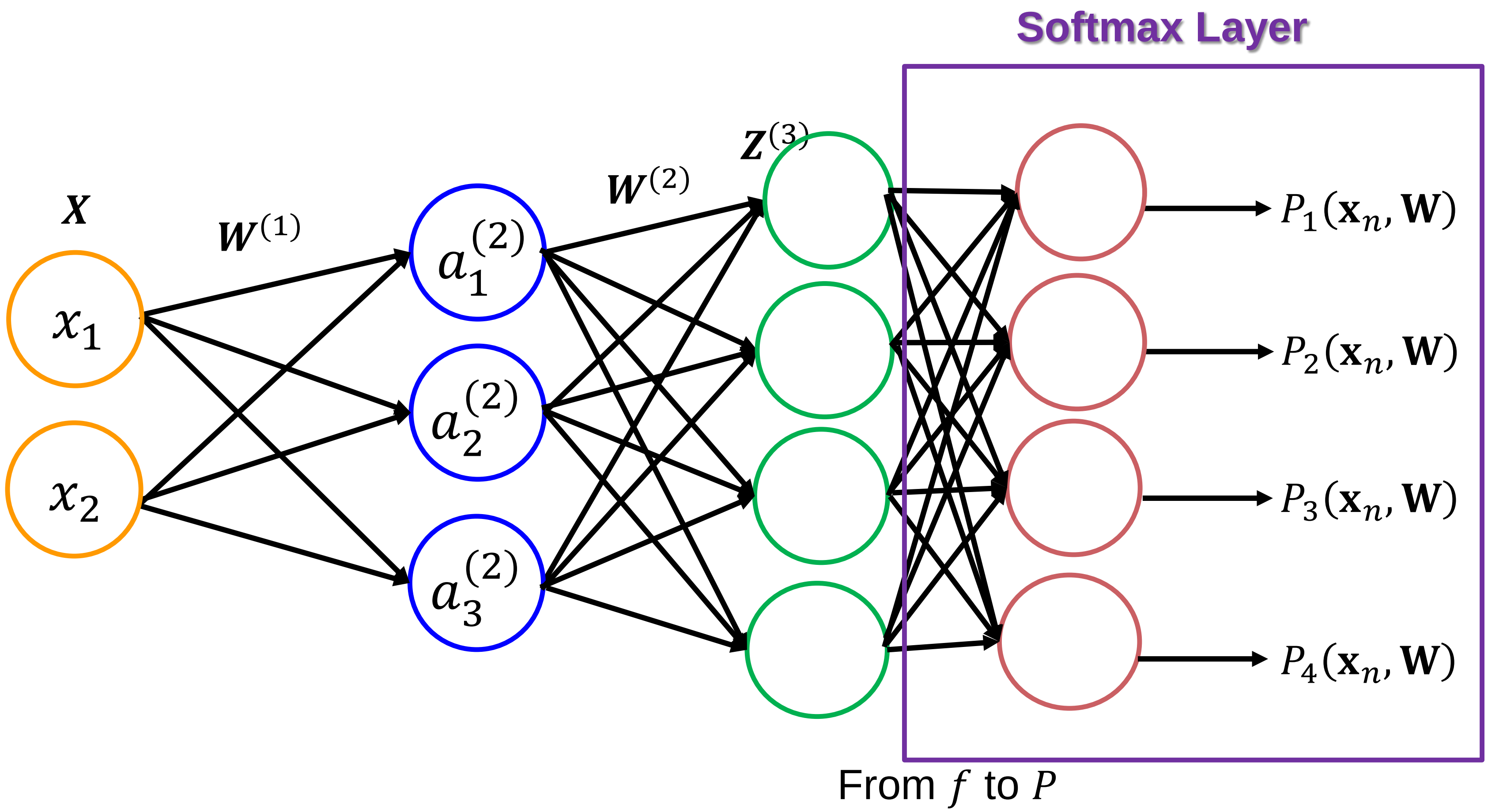
$$P_k(\mathbf{x}_n, \mathbf{W}) = \frac{e^{f_k(\mathbf{x}_n, \mathbf{W})}}{e^{f_1(\mathbf{x}_n, \mathbf{W})} + e^{f_2(\mathbf{x}_n, \mathbf{W})} + \cdots + e^{f_K(\mathbf{x}_n, \mathbf{W})}}, \qquad k = 1, 2, , \ldots, K$$

- This layer is called softmax layer

- Then the cross-entropy loss is defined as

$$L(\mathbf{W}) = -\frac{1}{N}\left[\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \log\bigl(P_k(\mathbf{x}_n, \mathbf{W})\bigr)\right]$$

Softmax Layer ($K = 4$)

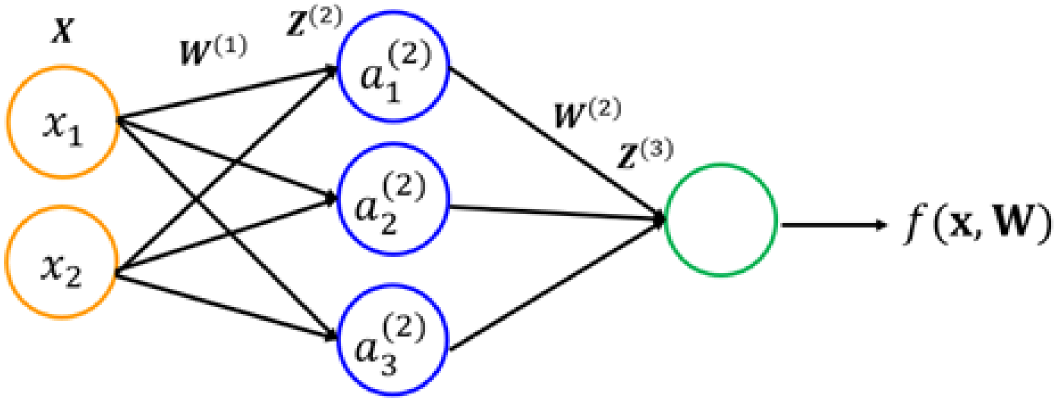


Neural Network Training

Parameters to be estimated

Target: minimize the loss function by changing the weights $\boldsymbol{W}^{(1)}$, $\boldsymbol{W}^{(2)}$

Let's still use 3 layer NN without bias units example that we had last week



$$\mathbf{W}^{(1)} = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} & w_{13}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} & w_{23}^{(1)} \end{bmatrix} \qquad \mathbf{W}^{(2)} = \begin{bmatrix} w_{11}^{(2)} \\ w_{21}^{(2)} \\ w_{31}^{(2)} \end{bmatrix}$$

Dimensionality

Why not simply try all the potential weights and see which combination produces the smallest loss?

Curse of dimensionality:

- 1 parameter: 1000 trials, 0.05 seconds
- 2 parameters: 1000 ×1000=1 million trials, 0.05*1000= 50 seconds
- 3 parameters: 1000 × 1000 × 1000=1 billion trials; (1,000,000*0.05)/(60*60)= 13.89 hours
- …
- 9 parameters: how long it will take?

Besides estimating the parameters of NN, other problems in neural network modelling:

- How to select the number of hidden layers?
- How to select the number of units in each hidden layer?
- How to perform feature selection?
- ...

Backpropagation Intuition

Backpropagation

- Backpropagation is a method used in artificial neural networks to calculate the error contribution of each neuron after a batch of data (in image recognition, multiple images) is processed. This is used by an enveloping optimization algorithm to adjust the weight of each neuron, completing the learning process for that case.

- Technically it calculates the gradient of the loss function. It is commonly used in the gradient descent optimization algorithm. It is also called **backward propagation of errors**, because the **error is calculated at the output and distributed back** through the network layers.

https://en.wikipedia.org/wiki/Backpropagation

- Phase 1: propagation. Each propagation involves the following steps:
  - Propagation forward through the network to generate the output value(s)
  - Calculation of the loss (error term)
  - Propagation of the output activations back through the network using the training pattern target in order to generate the **deltas (δ)** of all output and hidden neurons.

- Phase 2: weight (parameter) update. For each weight, the following steps must be followed:
  - The weight's output **deltas** and input activation are multiplied to find the gradient of the weight.
  - A ratio (learning rate) of the weight's gradient is subtracted from the weight.

Backpropagation

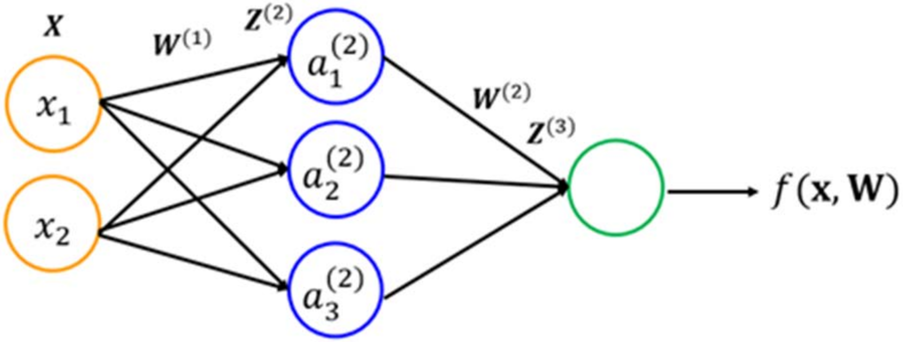
3 Layers Neural Network

- Now we are going to use back propagation to train this NN for regression
- The example we had last week
- No bias unit for simplicity

$$L(\mathbf{W}) = \frac{1}{2}\sum_{n=1}^{N}(f(\mathbf{x}_n, \mathbf{W}) - t_n)^2$$

Here N is omitted in $L(\boldsymbol{W})$ for simplicity. Will not change the optimization results.

x_1	x_2
1	0.875
0.25	1
0.75	0.25



-expense
1
0.4
0.6

Calculate Gradient

$$\frac{\partial L(\mathbf{W})}{\partial \mathbf{W}} = ?$$

Our parameter set $\boldsymbol{W}$ contains $\boldsymbol{W}^{(1)}$ and $\boldsymbol{W}^{(2)}$

$$\mathbf{W}^{(1)} = \left.\overbrace{\begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} & w_{13}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} & w_{23}^{(1)} \end{bmatrix}}^{\text{Hidden layer size}}\right\}\begin{array}{l}\text{Input layer} \\ \text{size.} \\ \text{2 features.}\end{array}$$

$$\mathbf{W}^{(2)} = \left.\overbrace{\begin{bmatrix} w_{11}^{(2)} \\ w_{21}^{(2)} \\ w_{31}^{(2)} \end{bmatrix}}^{\text{Output layer size}}\right\}\begin{array}{l}\text{Hidden} \\ \text{layer} \\ \text{size}\end{array}$$

$$\frac{\partial L(\mathbf{W})}{\partial \mathbf{W}^{(1)}} = \left.\overbrace{\begin{bmatrix} \frac{\partial L(\mathbf{W})}{\partial w_{11}^{(1)}} & \frac{\partial L(\mathbf{W})}{\partial w_{12}^{(1)}} & \frac{\partial L(\mathbf{W})}{\partial w_{13}^{(1)}} \\ \frac{\partial L(\mathbf{W})}{\partial w_{21}^{(1)}} & \frac{\partial L(\mathbf{W})}{\partial w_{22}^{(1)}} & \frac{\partial L(\mathbf{W})}{\partial w_{23}^{(1)}} \end{bmatrix}}^{\text{Hidden layer size}}\right\}\begin{array}{l}\text{Input} \\ \text{layer} \\ \text{size}\end{array}$$

$$\frac{\partial L(\mathbf{W})}{\partial \mathbf{W}^{(2)}} = \left.\overbrace{\begin{bmatrix} \frac{\partial L(\mathbf{W})}{\partial w_{11}^{(2)}} \\ \frac{\partial L(\mathbf{W})}{\partial w_{21}^{(2)}} \\ \frac{\partial L(\mathbf{W})}{\partial w_{31}^{(2)}} \end{bmatrix}}^{\text{Output layer size}}\right\}\begin{array}{l}\text{Hidden} \\ \text{layer} \\ \text{size}\end{array}$$

Backpropagation- Step 1

- First, work on the partial derivatives of loss with respect to $\boldsymbol{W}^{(2)}$.
- Note that now we start from the **"right"** of NN, or an opposite direction compared to forward propagation

$$\frac{\partial L(\mathbf{W})}{\partial \mathbf{W}^{(2)}} = \frac{\partial \frac{1}{2} \sum_{n=1}^{N} (f(\mathbf{x}_n, \mathbf{W}) - t_n)^2}{\partial \mathbf{W}^{(2)}}$$

Sum rules in differentiation: derivative of sum equals to the sum of derivatives

$$\frac{\partial L(\mathbf{W})}{\partial \mathbf{W}^{(2)}} = \sum_{n=1}^{N} \frac{\partial \frac{1}{2} (f(\mathbf{x}_n, \mathbf{W}) - t_n)^2}{\partial \mathbf{W}^{(2)}}$$

Remove summation for simplicity, will look after it later.
Also remove the subscript (n) in $f(\mathbf{x}_n, \mathbf{W})$ and t_n for simplicity.

$$\frac{\partial L(\mathbf{W})}{\partial \mathbf{W}^{(2)}} = \frac{\partial \frac{1}{2} (f(\mathbf{x}, \mathbf{W}) - t)^2}{\partial \mathbf{W}^{(2)}}$$

Backpropagation- Step 1

Derivative chain rule

$$\frac{\partial L(z(w))}{\partial w} = \frac{\partial L(z)}{\partial z} * \frac{\partial z(w)}{\partial w}$$

special meaning

t is a constant w.r.t. $\boldsymbol{W}^{(2)}$

$f(\mathbf{x}, \mathbf{W}) = \boldsymbol{\sigma}(\boldsymbol{z}^{(3)})$

$$\frac{\partial L(\mathbf{W})}{\partial \mathbf{W}^{(2)}} = \frac{\partial \frac{1}{2}(f(\mathbf{x}, \mathbf{W}) - t)^2}{\partial \mathbf{W}^{(2)}} = (f(\mathbf{x}, \mathbf{W}) - t)\frac{\partial f(\mathbf{x}, \mathbf{W})}{\partial \mathbf{W}^{(2)}}$$

$$= (f(\mathbf{x}, \mathbf{W}) - t)\boxed{\frac{\partial f(\mathbf{x}, \mathbf{W})}{\partial \boldsymbol{z}^{(3)}} \frac{\partial \boldsymbol{z}^{(3)}}{\partial \mathbf{W}^{(2)}}}$$

Derivative chain rule. Based on FP, $z^{(3)}$ is between$f(\mathbf{x}, \mathbf{W})$and $W^{(2)}$

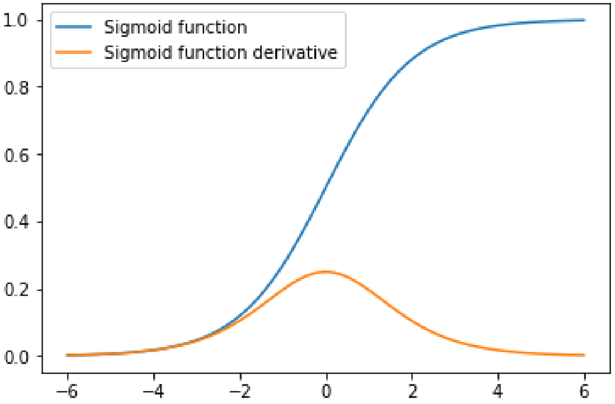
$$= (f(\mathbf{x}, \mathbf{W}) - t)\boxed{\frac{\partial \boldsymbol{\sigma}(\boldsymbol{z}^{(3)})}{\partial \boldsymbol{z}^{(3)}}} \frac{\partial \boldsymbol{z}^{(3)}}{\partial \mathbf{W}^{(2)}}$$

How to calculate this?

Sigmoid function derivative

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

$$\sigma'(z) = \frac{d}{dz}\left(\frac{1}{1+e^{-z}}\right) = \frac{e^{-z}}{(1+e^{-z})^2} = \sigma(z)(1 - \sigma(z))$$



$$\frac{\partial \boldsymbol{\sigma}(\mathbf{z}^{(3)})}{\partial \boldsymbol{z}^{(3)}} = \boldsymbol{\sigma}'\left(\boldsymbol{z}^{(3)}\right) = \boldsymbol{\sigma}(\boldsymbol{z}^{(3)})(1 - \boldsymbol{\sigma}(\boldsymbol{z}^{(3)}))$$

$$\frac{\partial L(\mathbf{W})}{\partial \mathbf{W}^{(2)}} = (f(\mathbf{x_1}, \mathbf{W}) - t_1)\boldsymbol{\sigma}\left(\boldsymbol{z}_{\mathbf{11}}^{(3)}\right)\frac{\partial \boldsymbol{z}_{11}^{(3)}}{\partial \mathbf{W}^{(2)}} + (f(\mathbf{x_2}, \mathbf{W}) - t_2)\boldsymbol{\sigma}\left(\boldsymbol{z}_{21}^{(3)}\right)\frac{\partial \boldsymbol{z}_{21}^{(3)}}{\partial \mathbf{W}^{(2)}} + (f(\mathbf{x_3}, \mathbf{W}) - t_3)\boldsymbol{\sigma}'\left(\boldsymbol{z}_{31}^{(3)}\right)\frac{\partial \boldsymbol{z}_{31}^{(3)}}{\partial \mathbf{W}^{(2)}}$$

For three data

$$\frac{\partial \boldsymbol{z}_{11}^{(3)}}{\partial \mathbf{W}^{(2)}} = \left[a_{11}^{(2)} \quad a_{12}^{(2)} \quad a_{13}^{(2)}\right]^{\mathrm{T}}, \qquad \frac{\partial \boldsymbol{z}_{21}^{(3)}}{\partial \mathbf{W}^{(2)}} = \left[a_{21}^{(2)} \quad a_{22}^{(2)} \quad a_{23}^{(2)}\right]^{T}, \qquad \frac{\partial \boldsymbol{z}_{31}^{(3)}}{\partial \mathbf{W}^{(2)}} = \left[a_{31}^{(2)} \quad a_{32}^{(2)} \quad a_{33}^{(2)}\right]^{T}$$

Backpropagation- Step 1

Now we write it for all the data $\mathbf{X}$ and $\mathbf{t}$

$$\frac{\partial L(\mathbf{W})}{\partial \mathbf{W}^{(2)}} = \underset{\text{Known}}{\boxed{(f(\mathbf{X},\mathbf{W}) - \mathbf{t})\sigma'(\mathbf{Z}^{(3)})}} * \boxed{\frac{\partial \mathbf{Z}^{(3)}}{\partial \mathbf{W}^{(2)}}}$$

How to calculate this?

Suppose $N = 3$:

$$\mathbf{Z}^{(3)} = \begin{bmatrix} z_{11}^{(3)} \\ z_{21}^{(3)} \\ z_{31}^{(3)} \end{bmatrix} \qquad a^{(2)} = \begin{bmatrix} a_{11}^{(2)} & a_{12}^{(2)} & a_{13}^{(2)} \\ a_{21}^{(2)} & a_{22}^{(2)} & a_{23}^{(2)} \\ a_{31}^{(2)} & a_{32}^{(2)} & a_{33}^{(2)} \end{bmatrix} \qquad \mathbf{W}^{(2)} = \begin{bmatrix} w_{11}^{(2)} \\ w_{21}^{(2)} \\ w_{31}^{(2)} \end{bmatrix}$$

$$\underset{3 \times 1}{\mathbf{Z}^{(3)}} = \underset{3 \times 3}{\boxed{a^{(2)}}} \underset{3 \times 1}{\mathbf{W}^{(2)}}$$

$\boldsymbol{Z}^{(3)}$ is actually a linear combination of $\boldsymbol{W}^{(2)}$ with $\boldsymbol{a}^{(2)}$, so:

$$\frac{\partial \mathbf{Z}^{(3)}}{\partial \mathbf{W}^{(2)}} = \begin{array}{cc} & \xrightarrow{\text{W}} \\ \downarrow \text{data} & \begin{bmatrix} a_{11}^{(2)} & a_{12}^{(2)} & a_{13}^{(2)} \\ a_{21}^{(2)} & a_{22}^{(2)} & a_{23}^{(2)} \\ a_{31}^{(2)} & a_{32}^{(2)} & a_{33}^{(2)} \end{bmatrix} \end{array}$$

Backpropagation- Step 1

$$\frac{\partial L(\mathbf{W})}{\partial \mathbf{W}^{(2)}} = \boxed{(\overset{3\times 1}{f(\mathbf{X},\mathbf{W}) - \mathbf{t}}) .* \overset{3\times 1}{\sigma'(\mathbf{z}^{(3)})}} * \boxed{\overset{3\times 3}{\frac{\partial \mathbf{Z}^{(3)}}{\partial \mathbf{W}^{(2)}}}}$$

????????

Here we use matrix element wise product

$$\boxed{\begin{bmatrix} f(\mathbf{x}_1,\mathbf{W}) - t_1 \\ f(\mathbf{x}_2,\mathbf{W}) - t_2 \\ f(\mathbf{x}_3,\mathbf{W}) - t_3 \end{bmatrix} .* \begin{bmatrix} \sigma'(z_{11}^{(3)}) \\ \sigma'(z_{21}^{(3)}) \\ \sigma'(z_{31}^{(3)}) \end{bmatrix}}$$

$$\Downarrow$$

Backpropagation error

$$\boxed{\delta^{(3)} = \begin{bmatrix} \delta_1^{(3)} \\ \delta_2^{(3)} \\ \delta_3^{(3)} \end{bmatrix}}$$

$$\boxed{a^{(2)} = \begin{bmatrix} a_{11}^{(2)} & a_{12}^{(2)} & a_{13}^{(2)} \\ a_{21}^{(2)} & a_{22}^{(2)} & a_{23}^{(2)} \\ a_{31}^{(2)} & a_{32}^{(2)} & a_{33}^{(2)} \end{bmatrix}}$$

Now we need to calculate $\delta^{(3)}$ times $a^{(2)}$

Backpropagation- Step 1

Taking transpose of $a^{(2)}$, then times $\left(a^{(2)}\right)^T$ by $\delta^{(3)}$

$$\left(a^{(2)}\right)^T = \begin{bmatrix} a_{11}^{(2)} & a_{21}^{(2)} & a_{31}^{(2)} \\ a_{12}^{(2)} & a_{22}^{(2)} & a_{32}^{(2)} \\ a_{13}^{(2)} & a_{23}^{(2)} & a_{33}^{(2)} \end{bmatrix} \qquad \delta^{(3)} = \begin{bmatrix} \delta_1^{(3)} \\ \delta_2^{(3)} \\ \delta_3^{(3)} \end{bmatrix}$$

3 × 3 3 × 1

Step 1: summary

1st equation uses matrix product

2nd equation uses matrix **element wise** product

3 × 1

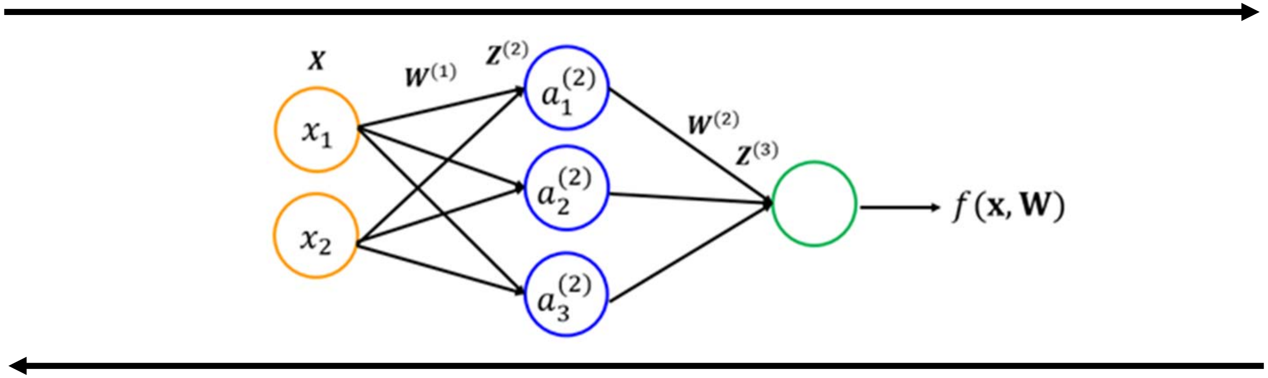
$$\frac{\partial L(\mathbf{W})}{\partial \mathbf{W}^{(2)}} = \boxed{\left(a^{(2)}\right)^T \delta^{(3)}}$$

$$\delta^{(3)} = (f(\mathbf{X}, \mathbf{W}) - \mathbf{t}).* \sigma'\left(\mathbf{Z}^{(3)}\right)$$

- We removed summation from loss function calculation previously, but now matrix multiplication also does the sum job.
- In the real implementation, first we calculate $\delta^{(3)}$, then calculate $\frac{\partial L(W)}{\partial W^{(2)}}$

Backpropagation

Forward propagation



Backpropagation

$$\delta^{(3)} = \begin{bmatrix} \delta_1^{(3)} \\ \delta_2^{(3)} \\ \delta_3^{(3)} \end{bmatrix}$$

- Given training examples, first run a "forward propagation" to compute all the activations throughout the network, including the output value of the NN$f(\mathbf{X}, \mathbf{W})$
- Then, for each node i in layer l, compute an "error term" $\delta_i^{(l)}$ that measures how much that node was "**responsible**" for any errors in our output

Backpropagation- Step 2

$$\mathbf{W}^{(1)} = \left.\overbrace{\begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} & w_{13}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} & w_{23}^{(1)} \end{bmatrix}}^{\text{Hidden layer size}}\right\}\text{Input unit size}$$

$$\frac{\partial \mathrm{L}(\mathbf{W})}{\partial \mathbf{W}^{(1)}} = \begin{bmatrix} \frac{\partial \mathrm{L}(\mathbf{W})}{\partial w_{11}^{(1)}} & \frac{\partial \mathrm{L}(\mathbf{W})}{\partial w_{12}^{(1)}} & \frac{\partial \mathrm{L}(\mathbf{W})}{\partial w_{13}^{(1)}} \\ \frac{\partial \mathrm{L}(\mathbf{W})}{\partial w_{21}^{(1)}} & \frac{\partial \mathrm{L}(\mathbf{W})}{\partial w_{22}^{(1)}} & \frac{\partial \mathrm{L}(\mathbf{W})}{\partial w_{23}^{(1)}} \end{bmatrix}$$

How to calculate this?

Backpropagation- Step 2

Let’s follow the similar process as step 1

$$\frac{\partial L(\mathbf{W})}{\partial \mathbf{W}^{(1)}} = \frac{\partial \frac{1}{2}(f(\mathbf{x},\mathbf{W})-t)^2}{\partial \mathbf{W}^{(1)}} = (f(\mathbf{x},\mathbf{W})-t)\frac{\partial f(\mathbf{x},\mathbf{W})}{\partial \mathbf{W}^{(1)}}$$

$$= (f(\mathbf{x},\mathbf{W})-t)\frac{\partial f(\mathbf{x},\mathbf{W})}{\partial \boldsymbol{z}^{(3)}}\frac{\partial \boldsymbol{z}^{(3)}}{\partial \mathbf{W}^{(1)}}$$

Note the difference compared to step 1

$$= (f(\mathbf{x},\mathbf{W})-t)\frac{\partial \boldsymbol{\sigma}(\boldsymbol{z}^{(3)})}{\partial \boldsymbol{z}^{(3)}}\frac{\partial \boldsymbol{z}^{(3)}}{\partial \mathbf{W}^{(1)}}$$

We calculated this in step 1

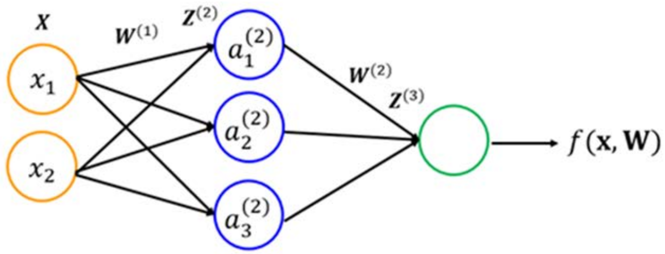
$$= \boxed{(f(\mathbf{x},\mathbf{W})-t)\boldsymbol{\sigma}'\left(\boldsymbol{z}^{(3)}\right)}\frac{\partial \boldsymbol{z}^{(3)}}{\partial \mathbf{W}^{(1)}}$$

$$= \boxed{\delta^{(3)}}\boxed{\frac{\partial \boldsymbol{z}^{(3)}}{\partial \mathbf{W}^{(1)}}}$$

How to calculate this?

Backpropagation- Step 2

Continued…



$$\frac{\partial L(\mathbf{W})}{\partial \mathbf{W}^{(1)}}$$

$$= \delta^{(3)} \boxed{\frac{\partial \boldsymbol{z}^{(3)}}{\partial \mathbf{W}^{(1)}}}$$

Derivative chain rule.
Based on FP, $W^{(2)}$, $a^{(2)}$, $z^{(2)}$ are between $z^{(3)}$and $W^{(1)}$

$$= \delta^{(3)} \frac{\partial \mathbf{z}^{(3)}}{\partial \mathrm{a}^{(2)}} \frac{\partial a^{(2)}}{\partial \mathbf{W}^{(1)}} = \delta^{(3)} \frac{\partial \mathbf{z}^{(3)}}{\partial \mathrm{a}^{(2)}} \frac{\partial a^{(2)}}{\partial \mathbf{z}^{(2)}} \frac{\partial \mathbf{z}^{(2)}}{\partial \mathbf{W}^{(1)}}$$

$$= \delta^{(3)} \left(\mathbf{W}^{(2)}\right)^{T} \frac{\partial a^{(2)}}{\partial \mathbf{z}^{(2)}} \frac{\partial \mathbf{z}^{(2)}}{\partial \mathbf{W}^{(1)}} = \delta^{(3)} \left(\mathbf{W}^{(2)}\right)^{T} \frac{\partial \sigma(\mathbf{z}^{(2)})}{\partial \mathbf{z}^{(2)}} \frac{\partial \mathbf{z}^{(2)}}{\partial \mathbf{W}^{(1)}}$$

$$= \delta^{(3)} \left(\mathbf{W}^{(2)}\right)^{T} .* \ \sigma'(\mathbf{z}^{(2)}) \boxed{\frac{\partial \mathbf{z}^{(2)}}{\partial \mathbf{W}^{(1)}}}$$

Use the same strategy as step 1 to calculate this.

THE UNIVERSITY OF SYDNEY

Backpropagation- Step 2

$$\frac{\partial L(\mathbf{W})}{\partial \mathbf{W}^{(1)}}$$

$$= \left(a^{(1)}\right)^T \left(\delta^{(3)}\left(\mathbf{W}^{(2)}\right)^T\right) .* \sigma'\left(\mathbf{z}^{(2)}\right)$$

For one data

$$a^{(1)} = \mathbf{x}$$

$$= \boxed{(\mathbf{x})^T} \underbrace{\left(\delta^{(3)}\left(\mathbf{W}^{(2)}\right)^T\right) .* \sigma'\left(\mathbf{z}^{(2)}\right)}_{\delta^{(2)}}$$

Step 2: summary for all data (e.g. N=3)

$$\overset{2 \times 3}{\frac{\partial L(\mathbf{W})}{\partial \mathbf{W}^{(1)}}} = \boxed{\overset{2 \times 3}{\mathbf{X}^T}\,\overset{3 \times 3}{\delta^{(2)}}}$$

$$\underset{3 \times 3}{\delta^{(2)}} = \left(\underset{3 \times 1}{\delta^{(3)}}\underset{1 \times 3}{\left(\mathbf{W}^{(2)}\right)^T}\right) .* \underset{3 \times 3}{\sigma'\left(\mathbf{Z}^{(2)}\right)}$$

- We removed summation from loss function calculation previously, but now matrix multiplication also does the sum job.
- In the real implementation, first we calculate $\delta^{(2)}$, then calculate $\frac{\partial L(W)}{\partial W^{(1)}}$

BP Step 1 + Step 2

Step 1: summary

$$\delta^{(3)} = (f(\mathbf{X},\mathbf{W}) - \mathbf{t}) .* \, \sigma'\left(\mathbf{Z}^{(3)}\right)$$

$$\frac{\partial L(\mathbf{W})}{\partial \mathbf{W}^{(2)}} = \left(a^{(2)}\right)^{T} \delta^{(3)}$$

Note the where the matrix element wise product “ .* ” is used

Step 2: summary

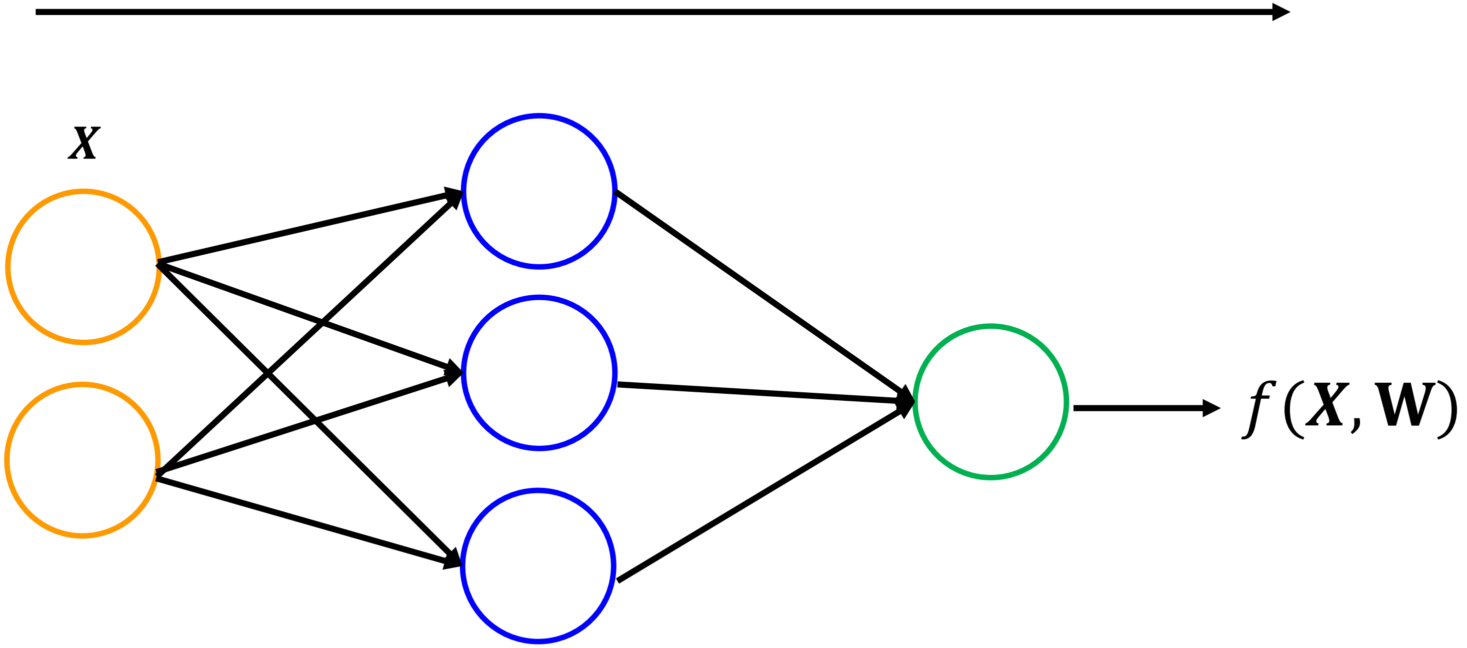
$$\delta^{(2)} = \left(\delta^{(3)}\left(\mathbf{W}^{(2)}\right)^{T}\right) .* \, \sigma'\left(\mathbf{Z}^{(2)}\right)$$

$$\frac{\partial L(\mathbf{W})}{\partial \mathbf{W}^{(1)}} = \mathbf{X}^{T}\delta^{(2)} = \left(a^{(1)}\right)^{T} \delta^{(2)}$$

Based on the layer index, we can see that this process can be easily repeated with more layers.

Backpropagation

Forward propagation



$\delta^{(2)}$ is a 3 × 3 matrix

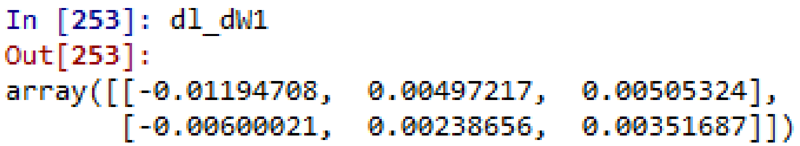
Backpropagation

Python Example

Calculate gradient

$$\frac{\partial L(\mathbf{W})}{\partial \mathbf{W}^{(1)}} = \begin{bmatrix} \frac{\partial L(\mathbf{W})}{\partial w_{11}^{(1)}} & \frac{\partial L(\mathbf{W})}{\partial w_{12}^{(1)}} & \frac{\partial L(\mathbf{W})}{\partial w_{13}^{(1)}} \\ \frac{\partial L(\mathbf{W})}{\partial w_{21}^{(1)}} & \frac{\partial L(\mathbf{W})}{\partial w_{22}^{(1)}} & \frac{\partial L(\mathbf{W})}{\partial w_{23}^{(1)}} \end{bmatrix}$$

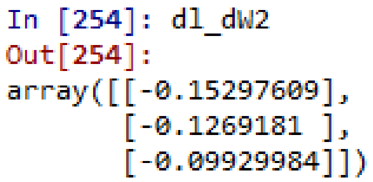
2 × 3



```
In [253]: dl_dW1
Out[253]:
array([[-0.01194708,  0.00497217,  0.00505324],
       [-0.00600021,  0.00238656,  0.00351687]])
```

$$\frac{\partial L(\mathbf{W})}{\partial \mathbf{W}^{(2)}} = \begin{bmatrix} \frac{\partial L(\mathbf{W})}{\partial w_{11}^{(2)}} \\ \frac{\partial L(\mathbf{W})}{\partial w_{21}^{(2)}} \\ \frac{\partial L(\mathbf{W})}{\partial w_{31}^{(2)}} \end{bmatrix}$$

3 × 1



```
In [254]: dl_dW2
Out[254]:
array([[-0.15297609],
       [-0.1269181 ],
       [-0.09929984]])
```

FP+BP (Regression)

Forward Propagation

$$a^{(1)} = \mathbf{X}$$

$$\mathbf{Z}^{(2)} = \mathbf{X}\mathbf{W}^{(1)}$$

$$a^{(2)} = \sigma(\mathbf{Z}^{(2)})$$

$$\mathbf{Z}^{(2)} = a^{(2)}\mathbf{W}^{(2)}$$

$$f(\mathbf{X}, \mathbf{W}) = a^{(2)} = \sigma(\mathbf{Z}^{(3)})$$

Backward Propagation

$$\delta^{(3)} = (f(\mathbf{X}, \mathbf{W}) - \mathbf{t}).* \, \sigma'\left(\mathbf{Z}^{(3)}\right)$$

$$\frac{\partial L(\mathbf{W})}{\partial \mathbf{W}^{(2)}} = \left(a^{(2)}\right)^{T} \delta^{(3)}$$

$$\delta^{(2)} = \left(\delta^{(3)}\left(\mathbf{W}^{(2)}\right)^{T}\right).* \, \sigma'\left(\mathbf{Z}^{(2)}\right)$$

$$\frac{\partial L(\mathbf{W})}{\partial \mathbf{W}^{(1)}} = \mathbf{X}^{T} \delta^{(2)} = \left(a^{(1)}\right)^{T} \delta^{(2)}$$

Calculate loss function

$$L(\mathbf{W}) = \frac{1}{2} \sum_{n=1}^{N} (f(\mathbf{x}_n, \mathbf{W}) - t_n)^2$$

Use “ .* ” to denote the element-wise product operator

FP+BP (Binary Classification)

Forward Propagation

$$a^{(1)} = \mathbf{X}$$

$$\mathbf{Z}^{(2)} = \mathbf{X}\mathbf{W}^{(1)}$$

$$a^{(2)} = \sigma(\mathbf{Z}^{(2)})$$

$$\mathbf{Z}^{(2)} = a^{(2)}\mathbf{W}^{(2)}$$

$$f(\mathbf{X},\mathbf{W}) = a^{(2)} = \sigma(\mathbf{Z}^{(3)})$$

sigmoid

Backward Propagation

$$\delta^{(3)} = (f(\mathbf{X},\mathbf{W}) - \mathbf{t})$$ t in 1 or 0 code

$$\frac{\partial L(\mathbf{W})}{\partial \mathbf{W}^{(2)}} = \left(a^{(2)}\right)^{T}\delta^{(3)}$$

$$\delta^{(2)} = \left(\delta^{(3)}\left(\mathbf{W}^{(2)}\right)^{T}\right) .* \ \sigma'\left(\boldsymbol{Z}^{(2)}\right)$$

$$\frac{\partial L(\mathbf{W})}{\partial \mathbf{W}^{(1)}} = \mathbf{X}^{T}\delta^{(2)} = \left(a^{(1)}\right)^{T}\delta^{(2)}$$

Calculate (logistic) loss function (1/N removed)

$$L(\mathbf{W}) = -\left[\sum_{n=1}^{N}\left(t_n \log\left(f(\mathbf{x}_n,\mathbf{W})\right) + (1-t_n)\log\left(1 - f(\mathbf{x}_n,\mathbf{W})\right)\right)\right]$$

Use “ .* ” to denote the element-wise product operator

Train a NN

The following is pseudocode of training our three-layer network (only one hidden layer):

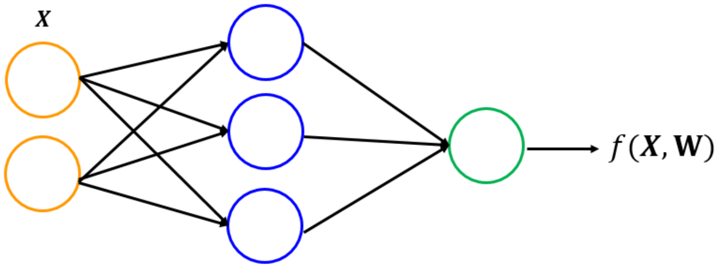
- Randomly initialize weights $\boldsymbol{W}^{(1)}$, $\boldsymbol{W}^{(2)}$
- Iterate the following procedure until stopping criterion satisfied:
  1. Implement the **forward propagation**
  2. Calculate the loss
  3. Implement the **backpropagation** and compute **partial derivatives** $\frac{\partial L(W)}{\partial W^{(2)}}, \frac{\partial L(W)}{\partial W^{(1)}}$
  4. Use below **gradient descent** to update weights $\boldsymbol{W}^{(1)}$, $\boldsymbol{W}^{(2)}$
  5. Incorporate the updated weights into the step 1

$$\mathbf{W}^{(1)} := \mathbf{W}^{(1)} - \alpha \frac{\partial \mathrm{L}(\mathbf{W})}{\partial \mathbf{W}^{(1)}}$$

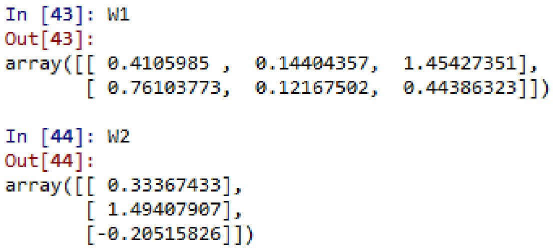
$$\mathbf{W}^{(2)} := \mathbf{W}^{(2)} - \alpha \frac{\partial \mathrm{L}(\mathbf{W})}{\partial \mathbf{W}^{(2)}}$$

Python example

```
# initialize the weights
input_layer_size= 2
hidden_layer_size= 3
output_layer_size= 1

# weight parameters
# define W(1): layer 1 to layer 2
np.random.seed(0)
W1= np.random.randn(input_layer_size, hidden_layer_size)
# define W(2): layer 2 to layer 3
W2= np.random.randn(hidden_layer_size, output_layer_size)
```

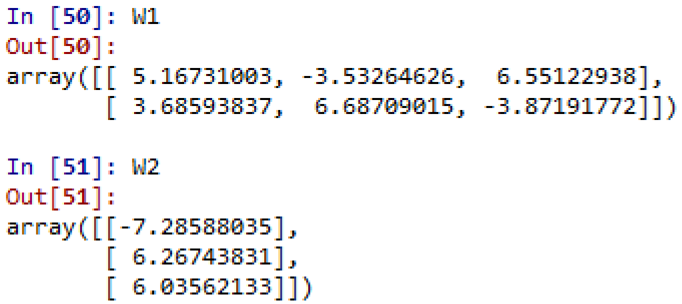


Randomly initialized weights



```
In [43]: W1
Out[43]:
array([[ 0.4105985 ,  0.14404357,  1.45427351],
       [ 0.76103773,  0.12167502,  0.44386323]])

In [44]: W2
Out[44]:
array([[ 0.33367433],
       [ 1.49407907],
       [-0.20515826]])
```

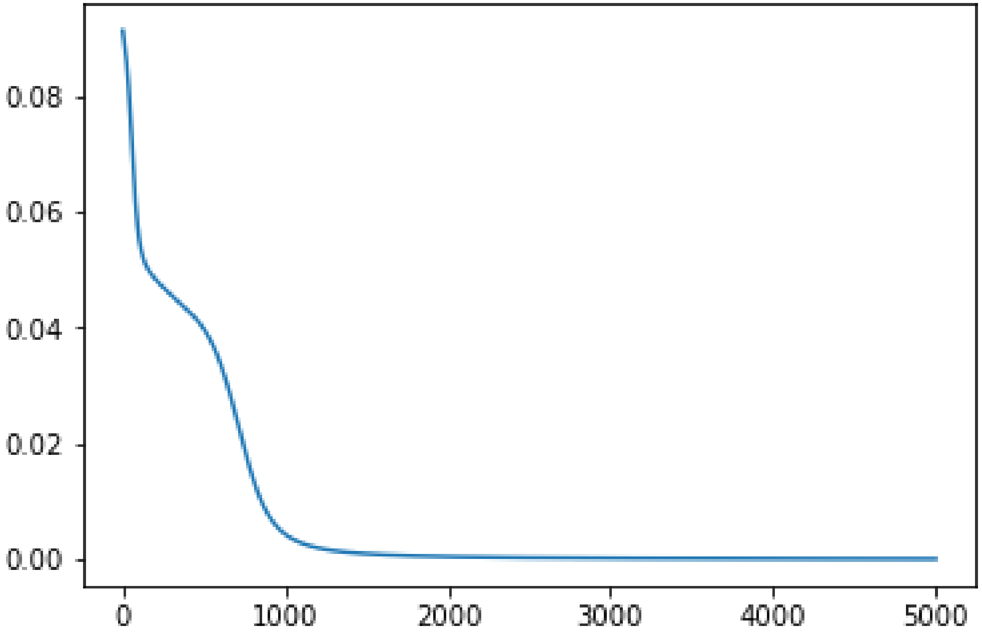
Updated weights after running the algorithm by 5000 iterations



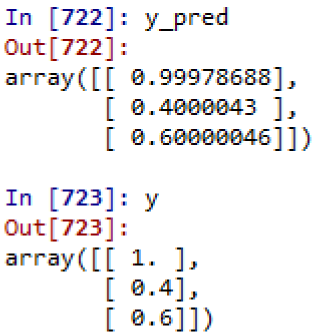
```
In [50]: W1
Out[50]:
array([[ 5.16731003, -3.53264626,  6.55122938],
       [ 3.68593837,  6.68709015, -3.87191772]])

In [51]: W2
Out[51]:
array([[-7.28588035],
       [ 6.26743831],
       [ 6.03562133]])
```

Python example

Loss function plot using gradient descent. numIterations= 5000.



Predicted y and actual y



```
In [722]: y_pred
Out[722]:
array([[ 0.99978688],
       [ 0.4000043 ],
       [ 0.60000046]])

In [723]: y
Out[723]:
array([[ 1. ],
       [ 0.4],
       [ 0.6]])
```

Regularization

NN Regression with Regularization

Can we keep have more and more neurons and hidden layers in the NN to improve the fitting on the training data?

- L: total number of layers of the NN
- S_l: number of units (not including bias unit) in the NN layer l
- One output unit

$$L(\mathbf{W}) = \frac{1}{2N}\sum_{n=1}^{N}(f(\mathbf{x}_n, \mathbf{W}) - t_n)^2 + \frac{\lambda}{2N}\sum_{l=1}^{L-1}\sum_{i=1}^{S_l}\sum_{j=1}^{S_{l+1}}\left(w_{ij}^{(l)}\right)^2$$

- How to select the number of hidden layers?
- How to select the number of units in each hidden layer?
- How to perform feature selection?
- ...

We can still incorporate the model selection techniques to choose the best model, including training & validation & test sets, CV, etc.

Classification with Regularization

- L: total number of layers of the NN
- S_l: number of units (not including bias unit) in the NN layer l
- One output unit (binary classification)

$$L(\mathbf{W}) = -\frac{1}{N}\left[\sum_{n=1}^{N}\Big(t_n \log\big(f(\mathbf{x}_n, \mathbf{W})\big) + (1 - t_n) \log\big(1 - f(\mathbf{x}_n, \mathbf{W})\big)\Big)\right] + \frac{\lambda}{2N}\sum_{l=1}^{L-1}\sum_{i=1}^{S_l}\sum_{j=1}^{S_{l+1}}\left(w_{ij}^{(l)}\right)^2$$